IPAC - An Interactive Approach to Access Control for Semi-Structured Data

Sriram Mohan *
Computer Science Department
Indiana University
Bloomington,USA
srmohan@cs.indiana.edu

Yuqing Wu *
School of Informatics
Indiana University
Bloomington,USA
yuqwu@indiana.edu

ABSTRACT

We propose IPAC (**I**nteractive a**P**proach to **A**ccess **C**ontrol for semi-structured data), a framework for XML access constraint specification and security view selection. IPAC clearly demarcates access constraint specification, access control strategy and security mechanism (implementation). It features a declarative access constraint specification language, a global access control strategy configuration unit, and an automatic security view generation and ranking tool. IPAC is the first system that assists the DBA in specifying access control strategies and access constraints on XML data, and helps the DBA in choosing the optimal plan that implements the specified strategy and access constraints accurately and efficiently.

1. INTRODUCTION

The research focus on XML repositories is switching from providing efficient storage and query processing techniques to general data management issues, such as access control. However, existing techniques [1, 2, 3, 5, 10] are limited to hiding nodes and subtrees with a few exceptions [4]. We have introduced ACXESS a query rewrite based access control model for XML [8, 9] that is capable of dealing with structural relationships. ACXESS utilized a graph editing language to express access constraints and enforces access control by rewriting user queries in XPath to XQuery with the guarantee that the users see only the information that they are allowed to see. However, these techniques for access control have significant shortcomings: either the access constraints are enforced via materialized views [4] and the performance is sacrificed, or the expressiveness of the security view is implemented by complicated procedural specifications that rearrange XML documental structures [8, 9], with the DBAs bearing the burden of specifying access constraints correctly and efficiently.

EXAMPLE 1. *Consider the management information of a research company. A highly simplified version of the XML structure is shown in Figure 1(a). Several access control levels can be envisioned on the schema. The company provides the 'public' with access to the lab's name, directors, all the research topics and a brief description, and also to the employees names and phone numbers, but the 'public' cannot find out the employees who work on a particular research topic. This requires that the parent-child relationship between* **research** *and* **employee** *elements be blocked.*

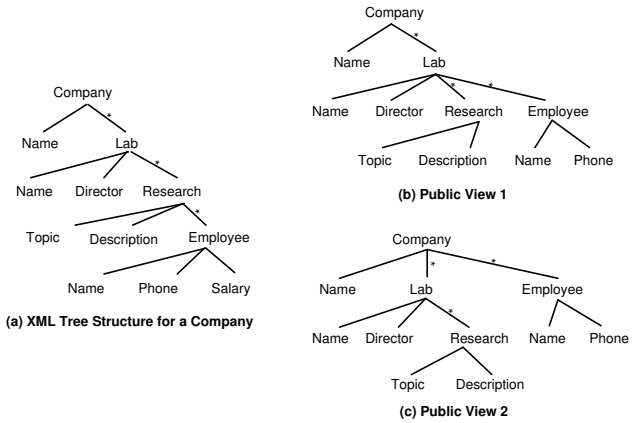

Figure 1: XML Tree and Security View Structures for Example 1

Existing approaches except [8, 9] either do not have the expressive power to extend access constraints to structural relationships [3, 10], or cannot implement them without view materialization [4] [1]. Moreover, all these techniques rely on DBAs to specify the exact manner in which the access constraints are implemented as a security view, rather than what the security constraints should be. In reality, there are multiple views that can implement an access constraint. For example, the access constraints for the 'public' user group can be implemented via the security views shown in Figure 1(b) and (c). With arbitrarily complicated XML schema and access constraints, it is not realistic for DBAs to think about every possible implementation and decide on the best. It is critical to exploit the computational power of modern computer systems to explore all possible implementations and make suggestions to the DBAs.

The quality of the security views that implement an access constraint vary. For instance, if a lab has exactly one research topic,

*The authors were supported by a grant from the Indiana University Faculty Research Support Program (IU FRSP) 2006.

VLDB '06, September 12-15, 2006, Seoul, Korea.

[1] View materialization is prone to update inconsistencies and can be non-realistic in real life scenarios.

the information about the association between the research project and the employees can be inferred easily from security view1 (as shown in Figure 1(b)), but not so easily from security view2 (as shown in Figure 1(c)). The quality of a security view depends on the criteria that defines the goodness of a view. Using measurement criteria such as 'inference' as guidelines, the security views can be ranked and the top-K views be presented to DBAs as system suggestions.

Existing approaches cannot exploit multiple security views for an access constraint specification as the access control strategy is inherently tied to the access constraint specification and the implementation process. Hence, existing approaches usually support only one access control strategy(e.g. *open world*). Fixing the access control strategy brings additional challenges to the access constraint specification process. For example, if a limited 'public' view allows users to access only the names of the employees in the company data as shown in Figure1(a), under the *open world* strategy, the DBA has to explicitly block access to all other subtree branches on the path from the root to the **name** element, while under the *closed world* strategy, the DBA can simply grant access to the subtrees identified by the XPath expression **/Company/Lab/Research /Employee/Name**. Being able to support multiple access control strategies not only simplifies the access constraint specification process, but also provides an opportunity for access constraints to be specified more efficiently.

2. INTERACTIVE APPROACH TO ACCESS CONTROL

The difficulties, as illustrated in the example, can be resolved by taking advantage of two well known concepts in relational databases that have not been exploited in the area of XML security. (1) *The Power of the Declarative Language* - a declarative query language (such as SQL) specifies what the query should do rather than how to do it. This feature brings about logical and physical independence - a fundamental property of relational database systems and opens the doors for system managed query processing and optimization, which contribute greatly to the query performance of relational databases. (2) *Harvesting the Power of the Computer* - The computer is billions of times faster than the human brain in computation and is adept at complicated computations and exploring huge solution spaces in search of optimal or favorable solutions. This provides relational databases with the ability to analyze thousands of query execution plans and identify the optimal plan.

With these in mind, we propose IPAC, a framework for XML access control that features a declarative access constraint specification language, a global access control strategy configuration unit, and an automatic security view generation and ranking tool. The goal is to assist DBAs in specifying access control strategies and access constraints on XML data, and to help DBAs in choosing the optimal plan that implements the specified access control strategy and access constraints accurately and efficiently.

2.1 System Architecture

In IPAC, we clearly demarcate the notion of access control strategy, access constraints, and their implementation and enforcement. The core of IPAC is its unique declarative access constraint specification language, which allows the DBAs to specify *what* to hide/reveal, rather than *how* the XML document is to be rearranged as seen in existing systems [3, 4, 8, 9]. This not only eases the job of the DBAs, but also leaves room for the system to suggest the top solutions that satisfy the access constraints. The latter task is carried out by the candidate security view generation component, which takes the declarative access constraint specification, access control strategy in effect (configured by DBAs using the strategy configuration component), and the schema, if available, and generates all possible security views that satisfy the access constraints. The security views thus generated have different criteria of quality and effectiveness. IPAC is capable of ranking the candidate security views, based on a set of supported ranking criteria and/or other ranking criteria specified by the DBA. The infrastructure of IPAC is shown in Figure 2. In the figure, we present the candidate view generation and ranking as two separate steps. In IPAC they are combined to further improve the performance, by adopting a more aggressive searching algorithm that computes only the top-K candidate security views. Only one candidate security view is selected (by IPAC or the DBA), implemented and enforced during query evaluation.

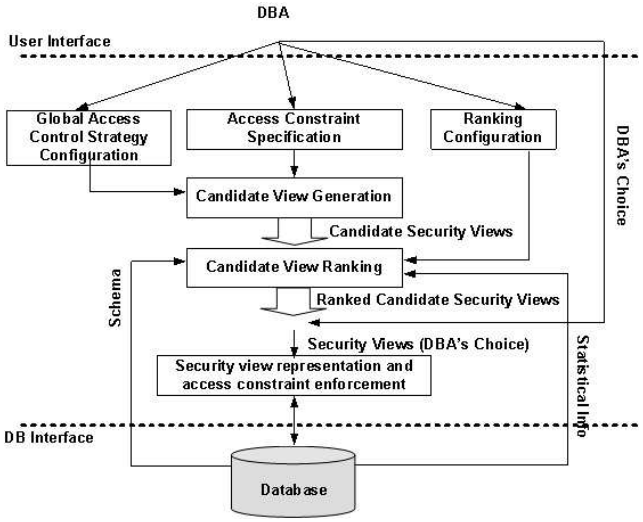

Figure 2: The Infrastructure of IPAC

The ACXESS system [8, 9] developed at Indiana University is used as the back-end for security view implementation and enforcement. IPAC utilizes the rewrite rules developed as a part of the ACXESS system [8, 9] to rewrite user XPath queries and enforce the specified access constraints. The focus of IPAC is to provide a clear demarcation between strategy and mechanism and to guarantee that the security view being implemented satisfies the desired access constraints and is optimal with respect to the selected quality criteria.

2.2 Declarative Access Constraint Specification Language

The goal of the declarative access constraint specification language is to assist the DBAs in specifying access constraints. It is specific enough to express access constraints that exist only in the context of XML and yet generic enough to accommodate different access control strategies. The key feature of the language is that it provides a declarative interface for access constraint specification and clearly separates the constraint specification from the (specific) implementation of such access constraints on a specific XML database. Some of the desired features enabled by this language include intuitive human interface, automatic security view generation and tuning, and high performance access constraint enforcement.

Primitive	Explanation
deny($destPath$)	deny/allow access to the subtrees that match $destPath$.
allow($destPath$)	
blockStruct($ancsPath$, $relPath$)	deny/allow access to the structural relationship between nodes that match $ancsPath$ and subtrees that match $relPath$.
allowStruct($ancsPath$, $relPath$)	
dissociate($ancsPath$, $relPath_1$, ..., $relPath_n$)	deny/allow access to the structural association relationship among subtrees that match $relPath_1$ to $relPath_n$, with respect to the common ancestor that matches to $ancsPath$.
associate($ancsPath$, $relPath_1$, ..., $relPath_n$)	

Table 1: Primitives in the Declarative Access Constraint Specification Language

The declarative language introduces a set of primitives that (i) specify the values (nodes and subtrees) to be revealed or hidden, (ii) constrict the path along which certain values can be accessed, and (iii) regulate the correlations among nodes and subtrees. A brief summary of the primitives is available in Table 1. The declarative language focuses on specifying $what$ to hide/reveal, rather than how to do so. It covers both values and structural relationships, which are equally important in the context of XML.

2.3 Global Access Control Strategy Configuration

Demarcation of strategy from implementation has been studied in relational databases [6, 7]. A major drawback of existing XML access control systems is that they have all been developed with a specific access control strategy in mind. This entails that the access constraints be specified in terms of the supported strategy. While this is trivial for some access constraints, specification of other access constraints may become quite complicated as shown in Example 1. IPAC supports several well-known strategy templates including (i) open world (ii) closed world (iii) denial takes precedence. Strategies thus selected determine the interpretation of the declarative access constraint specification, the derivation of authorizations, conflict resolution, and integrity constraint checking. The strategies combine with the declarative access constraint specification in determining the set of candidate security views that are generated.

2.4 Candidate Security View Generation

The flexibility of the declarative language in conjunction with the ability to support multiple strategies facilitates the generation of multiple candidate security views. In IPAC we introduce a search algorithm that searches the solution space of possible security views corresponding to the declarative access constraint specification and the access control strategy specified by the DBA. This algorithm consists of three passes: (i) a scan of the primitives in the access constraint set that annotates the value and structural relationship under effect in the XML schema; (ii) a top-down scan of the annotated XML schema tree that identifies the possible conflicts among the access constraint primitives; (iii) a bottom-up scan of the XML schema tree that resolves the conflicts and generates security views that satisfy the access constraints. This algorithm guarantees that all access constraints are accommodated. Using the notion of $subtree\ equivalence$, the number of candidate security views being generated is finite.

EXAMPLE 2. *Let us reconsider the access constraints for the 'public' view as described in Example 1. The declarative access constraint specifications under both open world and close world strategies are shown in Table 2. Multiple security views, including those shown in Figure 1(b) and (c), are generated for both specification/policy combinations.*

Strategy	Access Constraint Specification
open world	deny(/Company/Lab/Research/Salary) blockStruct(/Company/Lab/Research,Employee)
close world	allow(/Company) deny(/Company/Lab/Research/Salary) blockStruct(/Company/Lab/Research, Employee)

Table 2: Example Declarative Access Constraint Specifications

2.5 Candidate Security View Ranking

Even though all candidate views generated satisfy the access constraints, the quality and the effectiveness of the candidate security views vary, depending on the quality criteria. IPAC supports a set of criteria including (i) the amount of information available to the users, (ii) the complexity of the view construction, (iii) the complexity of the query rewrite process for access constraint enforcement, (iv) the quality of the rewritten query, and (v) potential information leakage (inference). A security view that favors one criterion may be forced to sacrifice on another. IPAC provides a ranking configuration tool that allows DBAs to prioritize the importance of these criteria. The candidate security views will be ranked accordingly. In addition, DBAs can define their own ranking criteria in the form of user defined functions. For clarity, the candidate security view generation and ranking are presented as two separate steps. In IPAC they are combined to further improve the performance, by adopting a more aggressive searching algorithm that prunes less attractive views early in the view generation process, searches a smaller solution space, and generate only the top-K candidate security views. Only one candidate security view is selected (by IPAC or the DBA) and is implemented and enforced using the ACXESS system [8, 9].

3. DEMONSTRATION PROPOSAL

We will demonstrate the expressive power of the declarative access constraint specification language, the quality of our security view suggestions to DBAs and the efficiency of our search algorithms. We will also demonstrate the impact of different strategies and ranking configurations on the results, which will further illustrate the value IPAC brings by combining an easy-to-use interactive interface with an efficient security implementation. In particular, we will demonstrate IPAC, focusing on the following features.

3.1 Access Constraint Specification and Candidate Security View Generation

The access constraint specification interface of IPAC visualizes the XML schema in the form of a schema tree. Access constraints are specified in a dialog box (on the bottom) and candidate security views are generated and also visualized in tree format (on the right side), as shown in Figure 3. During the demonstration, visitors are welcome to specify access constraints on any of the XML schema provided and examine the suggested candidate security views. Vis-

itors can also select the security view of their choice from the candidate views, and test the effectiveness of the access constraint enforcement by running queries and examining the results.

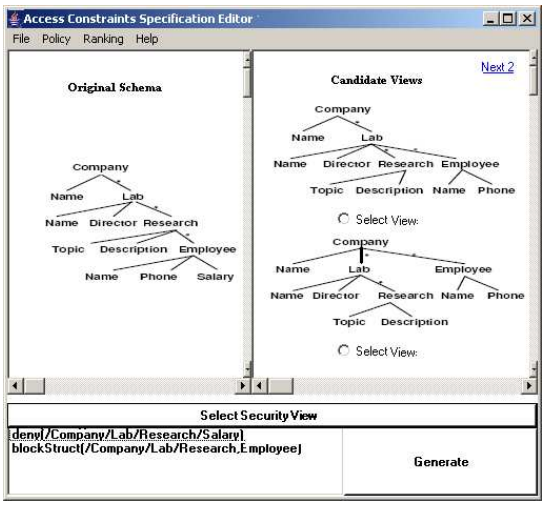

Figure 3: Demo: Access Constraint Specification and Candidate Security View Generation

3.2 Global Access Control Strategy Configuration

One of the major contributions of IPAC is that it provides a clear demarcation between the access control strategy and mechanism. For every access constraint specification, the DBA can trigger the strategy configuration tool (as shown in Figure 4) to make changes to the access control strategies in effect for the candidate view generation process. Open world and denial-takes-precedence are implemented in IPAC as the default strategies. We will showcase the strength and flexibility of IPAC by demonstrating the impact of the security strategies on the candidate view generation process. In addition, we will demonstrate how a security constraint specification can be specified differently based on different access control strategies in effect. The difference in complexity of the access constraint specifications will further illustrate the value of IPAC in supporting multiple security strategies.

Figure 4: Demo: Global Access Control Strategy Configuration

3.3 Ranking Candidate Security Views

Another significant feature of IPAC is that it provides a mechanism for ranking the candidate security views based on multiple ranking criteria. It also provides DBAs with the ability to be in control of the ranking process via its ranking configuration tool (as shown in Figure 5). Using this tool, DBAs can prioritize the ranking criteria and specify the number of suggestions (candidate security views) provided by the system. We will demonstrate the power of the ranking tool by showing different sets of top-K candidate security views in different ranking orders, for the same access constraint specification with different ranking configurations.

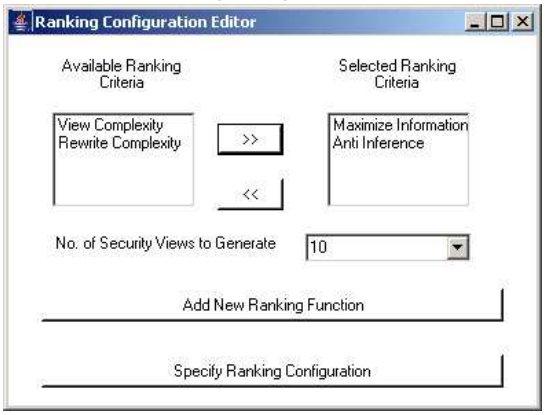

Figure 5: Demo: Ranking Configuration

4. SUMMARY

We introduce and demonstrate IPAC, the first system that assists DBAs in specifying access constraints on XML data and helps DBAs in choosing the optimal mechanism for implementing and enforcing such access constraints. IPAC features a declarative language for access constraint specification, supports multiple access control strategies, and provides efficient search and ranking algorithms to promote the accuracy and the performance of the access constraint specification process.

5. REFERENCES

[1] B. Carminati and E. Ferrari. AC-XML documents: improving the performance of a web access control module. In *SACMAT*, 2005.

[2] B. Carminati, E. Ferrari, and E. Bertino. Securing XML data in third-party distribution systems. In *CIKM*, 2005.

[3] W. Fan, C.-Y. Chan, and M. Garofalakis. Secure XML querying with security views. In *ACM SIGMOD*, 2004.

[4] B. Finance, S. Medjdoub, and P. Pucheral. The case for access control on XML relationships. In *CIKM*, 2005.

[5] I. Fundulaki and M. Marx. Specifying access control policies for XML documents with XPath. In *SACMAT*, 2004.

[6] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM TODS*, 26(2), 2001.

[7] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *IEEE Symposium on Security and Privacy*, 1997.

[8] S. Mohan, J. Klinginsmith, A. Sengupta, and Y. Wu. Access control for XML with enhanced security specifications. In *ICDE*, 2006.

[9] S. Mohan, A. Sengupta, and Y. Wu. Access control for XML - a dynamic query rewriting approach. In *CIKM*, 2005.

[10] N. Seki, M. Kudo, J. Myllmaki, and H. Pirahesh. A function-based access control model for XML databases. In *CIKM*, 2005.